Using NLP tools in the Specification Phase

Àngels Hernández & Núria Castell
TALP Research Center
Universitat Politècnica de Catalunya

Abstract

The software quality control is one of the main topics in the Software Engineering area. To put the effort in the quality control during the specification phase leads us to detect possible mistakes in an early steps and, easily, to correct them before the design and implementation steps start. In this framework the goal of SAREL system, a knowledge-based system, is twofold. On one hand, to help software engineers in the creation of quality Software Requirements Specifications. On the other hand, to analyze the correspondence between two different conceptual representations associated with two different Software Requirements Specification documents.

For the first goal, a set of NLP and Knowledge management tools is applied to obtain a conceptual representation that can be validated and managed by the software engineer.

For the second goal we have established some correspondence measures in order to get a comparison between two conceptual representations. This information will be useful during the interaction between the customer and the software engineer.

Keywords: *Natural language processing, Requirements Engineering, Knowledge-based systems, Software Quality.*

The specification phase is one of the most important and least supported parts of the software development process [1]. In this stage it is very important to control the quality of the specifications in order to detect possible mistakes as early as possible. In general, the earlier in the life cycle those potential errors are identified the easier it is to eliminate. This is the reason why developers increasingly try to identify possible mistakes in the early phases of software development.

During the initial phase, the software specifications are often written in natural language. Even after being formalized, this original documentation may also serve during later phases and during the functioning and maintenance of the developed computer system. The use of natural language has associated problems as ambiguity, inaccuracy and inconsistency [2]. Therefore it is important to control the writing of these documents because many more mistakes can be detected if the writing is clear and concise.

Documentation writing is guided by the norms that define the linguistic restrictions required to satisfy the specifications. These norms are of two types: those related to the use of natural language in general and those based on terminological restrictions linked to a particular domain (for example, the European Space Agency norms). Both of them restrict the use of natural language through a set of rules, which limit much kind of irregularities that occur during the interpretation of natural language.

In addition to linguistic restrictions, the norms also include software engineering constraints related to the software quality properties, among others: completeness, traceability, consistency, verifiability and modifiability.

Many projects have tackled the use of Natural Language in the specification phase, CREWS project is an important one. Among its publications we want to remark [3] where the scenario based approach and a linguistic based instrument have been proposed for improving requirement engineering tools and techniques. Within the framework of quality documents the ATTEMPTO approach [4] should be pointed out, whose main goal is to reduce ambiguity and vagueness inherent in NL. Another important work is QuARS [5] where a tool for the analysis of natural language software requirements based on a quality model is presented. Following these trends, we have developed a system that applies Artificial Intelligence tools to the Requirements Engineering.

SAREL System

SAREL (*Sistema d'Ajut a la Redacció d'Especificacions en Llenguatge Natural*) an Assistance System for Writing Software Specification in Natural Language, has been designed in order to obtain a high-quality Software

Requirements Specification (SRS). The aim is to assist the engineers in the creation of preliminary software specifications written in natural language. This assistance system is based on Knowledge Representation and Linguistic Tools and it leans on two main elements: the Knowledge Base (KB) and the Requirements Base (RB). The Knowledge Base contains a conceptual representation of the relevant concepts of the application domain, and the Requirements Base contains a conceptual representation of the Software Requirements Specification.

Taking into account the IEEE Standard [6], there are three essential sections in a SRS:

- Introduction, which provides an overview of the entire SRS

- The Overall Description, which describes the general factors that affect the product and its requirements.
- The Specific Requirements section, which contains all the software requirements, deepening into enough detail so as to enable engineers to design a system that satisfies those requirements.

We distinguish the Introduction and the Overall Description as sections that should be used in the Knowledge Base construction. Up to a certain point, these two sections contain all the background information needed to understand the problem as a whole. The Requirements Base represents the information contained in the Specific Requirements Section. In figure 1 we can see the Knowledge Base and the Requirements Base Generation Processes.
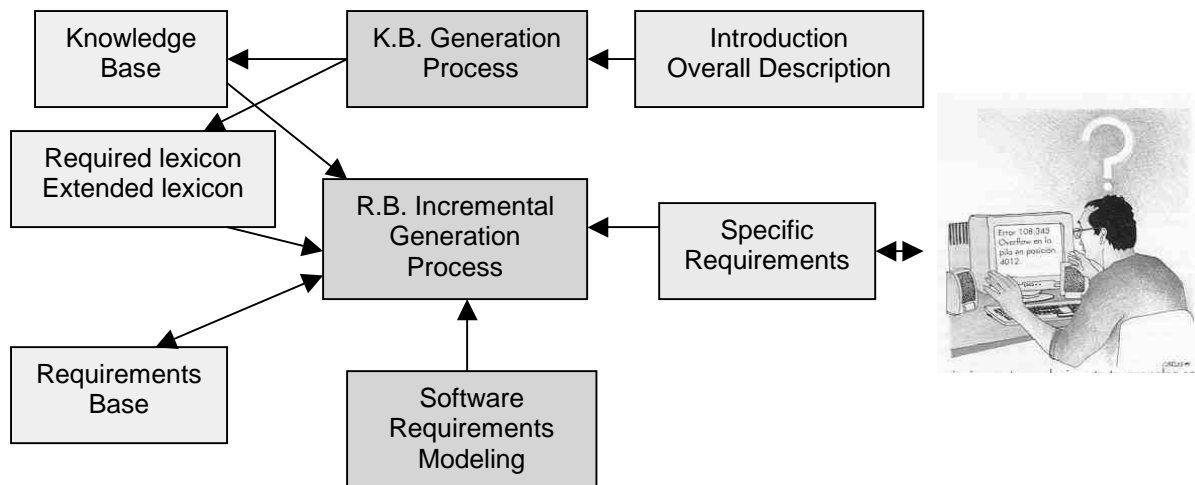

Figure 1. Knowledge Base and Requirements Base Generation

Knowledge Base Generation

We have implemented an automatic construction process that generates the Conceptual Representation of the Introduction and Overall Description. This process is split into three steps showed on the left side in Figure 2:

- The Lexicon Generator extracts from the original text the required lexicon and the extended lexicon using the Spanish semantic network Wordnet [7].
- The Lexical Refinement generates a new Introduction and a new Overall Description where all the words belong to the required lexicon (without synonyms).

- The KB Conceptual Generator generates a hierarchy of concepts grouped into two main classes: Objects and Activities.

The difference between the extended lexicon and the required lexicon is that the first lexicon contains all the words related to the application domain while the second is a subset of the extended. Each word included in the required lexicon is the representative that has been chosen from the synonyms set.

Lexicon Generator

To create the two lexica, we take profit of our general NLP tools (http://www.lsi.upc.es/~nlp). Firstly, the

Morphological Analyzer Maco+ [8] and the POS tagger Relax [9] process the sentences contained in the Introduction and the Overall Description of the document. The output of this process is a list of words with its corresponding PAROLE tag (www.tei-c.org/Applications).

Secondly, the system split the list into three different sublists corresponding with names, verbs and adjectives. The Synonym Analyzer (using Spanish Wordnet) processes each of them, in order to obtain the required and extended names, the required and extended verbs and the required and extended adjectives. In a first step, the Analyzer finds all the synsets and its corresponding label for each word in the text. For all synsets the system will find all the

synonyms associated. Third, the system finds the possible coincidences between the first word that belongs to the original text and the rest. To decide which word will be the representative of the synonym set the analyzer will consider the frequency of each word in the original text.

Lastly, grouping appropriately all the outputs supplied by the Synonyms Analyzer, we obtain the required lexicon and the extended lexicon. The existence of both lexica offers to the engineer some kink of writing flexibility, without to disobey the norms. The system will finally provided a document in accordance with these norms.
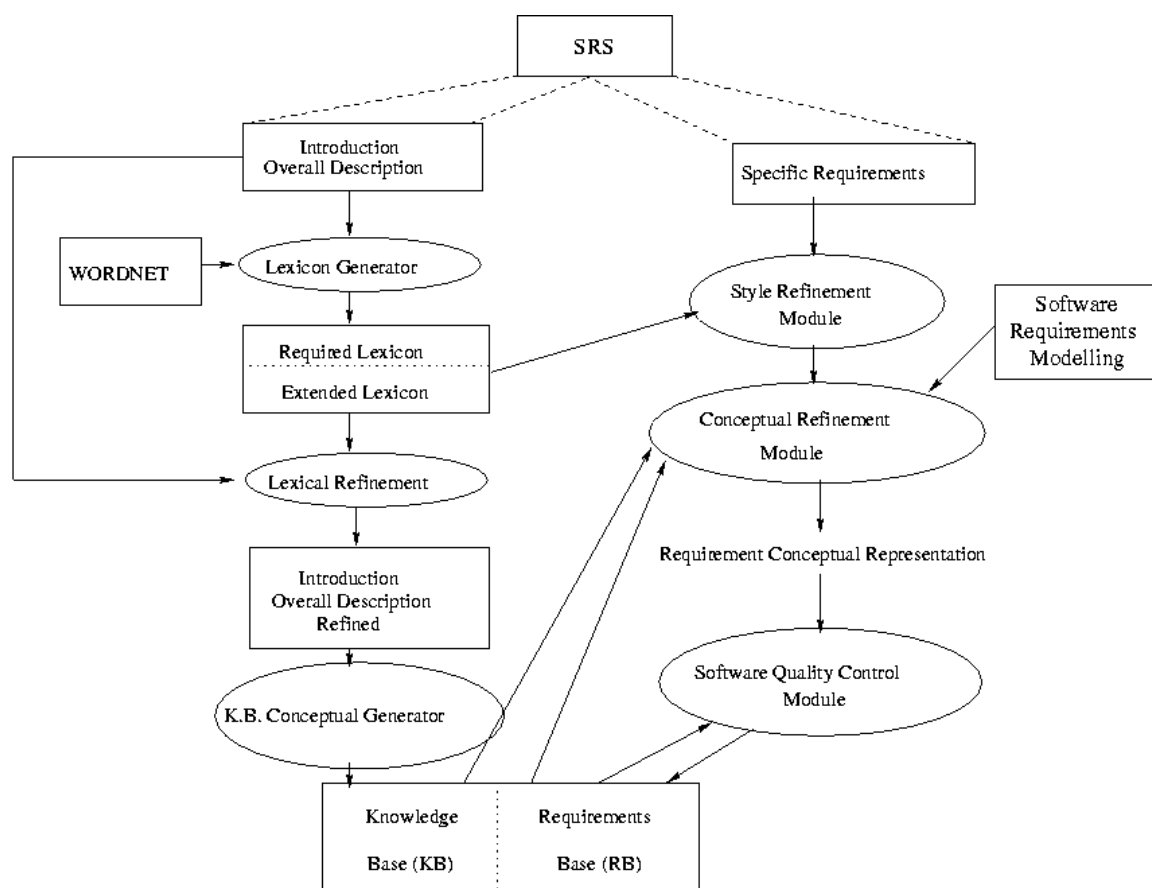

Figure 2. Modules of SAREL

Lexical Refinement

The following step is to refine the Introduction and the Overall Description in order to get a text without

synonyms. The output is a text where all the names, verbs and adjectives belong to the required lexicon. To get that, for each word we apply the following criteria: if the word belongs to the required lexicon, it

is added to the text refined; if the word belongs to the extended lexicon the Lexical Refinement processor substitutes this word by the representative of its synonym set. For a soundness replacement, the Morphological Analyzer (Maco+) is used in order to obtain the right conjugation.

We want to emphasize that the obtained output is a refined text without synonyms, containing the same information of the original text. The Lexical Refinement is an essential process taking into account that the refined text is the starting point to the creation of the Knowledge Base Conceptual Representation.

KB Conceptual Generator

From the refined text, the KB Conceptual Generator builds a hierarchy of concepts grouped into two main classes: Objects and Activities. This process is split into two steps: first, Object nodes generation, and second, Activity nodes generation.

For the creation of Object nodes we have establish the following rules that will be applied in a sequential way:

(a) Nodes corresponding to simple names tagged as: NC (Common Nouns)

(b) Nodes corresponding to noun phrases tagged as: {NC followed by NC} or {NC followed by AQ (Qualifying)} or {NC followed by VMP (Participle)}

(c) Nodes corresponding to the following schema: {A + *de* + B (A + of + B)}, where A and B are object nodes previously generated.

The rules established for the Activity nodes generation are:

(a) Nodes corresponding to simple verbs tagged as VMI (intransitive verbs) or {VMN (infinitive).

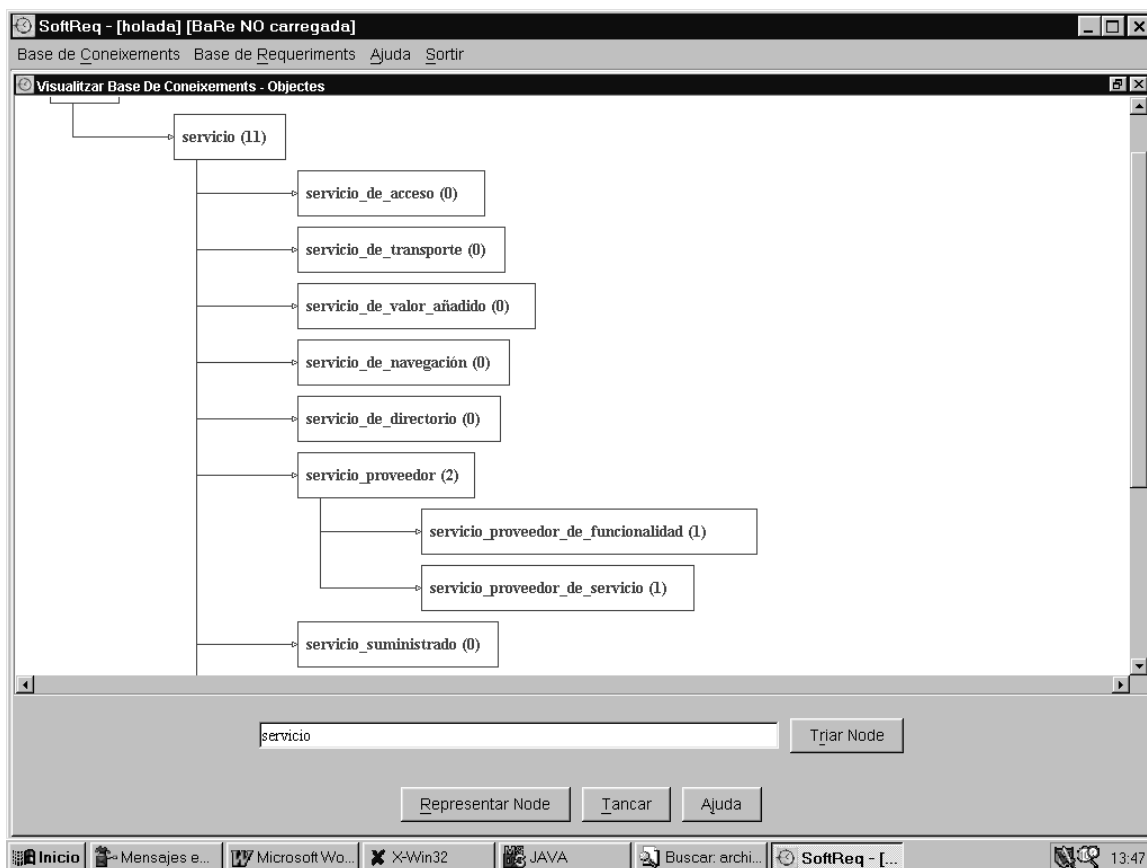(b) Nodes corresponding to verbal groups tagged as: {VMI followed by VMN}.

Figure 3. Knowledge Base Visualization

When the KB construction process has ended, the software engineer can visualize the Conceptual Representation obtained. In Figure 3 we present the hierarchy associated to the *servicio* (service) concept obtained from a Telecommunications Company document.

Requirements Base Generation

In relation to the Requirements Base, we have implemented a semi-automatic construction process that generates the Conceptual Representation of the RB. The requirements, grouped in the Specific Requirements section, can be classified in some general classes [6]. The only ones considered by SAREL at present are:
- Functional Requirements
- Performance Requirements
- External Interface Requirements

In order to control the set of software requirements contained in the Software Requirements Specification document, we have considered necessary to establish the different semantic roles (required and optional) associated to each kind of requirement [10].

- For Functional Requirements the semantic roles required are: Agent, Action and Patient. The conceptual representation associated with the functional requirement: "The transfer system shall transfer the clock-in to the communications server" is:

  (Define-node requirement-x
  (Agent transfer-system)
  (Action transfer)
  (Patient clock-in)
  (To-loc communications-server))

  Here To-loc is an optional semantic role.

- For Performance Requirements, the required semantic roles are: {Patient, Measurement, At-value, and Unit} or {Patient, Measurement, From-value, To-value, and Unit}. An example of performance requirement and its conceptual representation is: "An identifying card shall be made-up in less than 1 minute".

  (define-node requirement-y
  (Patient identifying-card)
  (Measurement making-up-time)
  (At-value 1)

  (Unit minutes))

- The required semantic roles for the External Interface Requirements are: {Patient and Qualitative-feature} or {Patient, Quantitative-feature, and Units-feature). An example of external interface requirement and its conceptual representation is: "The monitor of the personal computer will be colour and 15 inches"

  (define-node requirement-z
  (Patient monitor)
  (Qualitative-feature colour))

  (define-node requirement-t
  (Patient monitor)
  (Quantitative-feature 15)
  (Units-feature inches))

At present, we are reviewing and adding new semantic roles for each requirement class. The Conceptual Refinement Module will use this information in order to get the right conceptual representation associated to a given requirement.

The information represented in the Requirements Base corresponds to the information contained in the Specific Requirements section. The assistance process validates every requirement introduced by the engineer taking into account the writing norms and the quality properties. The controls are grouped into three modules presented, on the right, in Figure 2: the Style Refinement Module, the Conceptual Refinement Module and the Software Quality Control Module.

Style Refinement Module

The Style Refinement Module controls the requirement according to the writing norms and it is broken down into four steps: lexical analysis, syntactic-semantic analysis, ambiguity control and simplicity control.

The lexical analysis carries out the control of the lexicon used in the requirement. It verifies whether the words belong to the application domain lexicon or not. To do so, the system uses three different lexicons: the extended and required lexica, previously described, and a general lexicon containing general words as prepositions, conjunctions, etc.
The analyzer ensures all words contained in the requirements belong to one of the three lexicons. After that, all words only belonging to the extended

lexicon are substituted by synonyms from the required lexicon. The lexical validation process is independent of the application domain. Following we can see an example. The requirement is presented in Spanish with its corresponding translation in English.

Spanish: El sistema AEROS manipulará el ordenador de a bordo y el estado del vehículo espacial.

English: The AEROS system shall manipulate the computer on board and the status of the space-vehicle.

The lexical analysis classifies the blue words as general, the green words as required and the red words as extended. After the lexical analysis the extended words have been replaced by required words.

Spanish: El sistema AEROS controlará el ordenador de a bordo y el estado del vehículo espacial.

English: The AEROS system shall control the computer on board and the status of the space-vehicle.

The syntactic-semantic analysis applies a set of tools, developed in our group, in order to get a syntactic representation associated to the introduced requirement:
- The Morphological analyzer Maco+ [8], tokenizes the text and produces as output all morphological interpretations possible for each token.
- The POS tagger Relax [9] selects the right POS and lemma for each word in the given context.
- The syntactic chart-based Parser Tacat takes as input the output of the POS tagger Relax, and produces a syntactic-semantic representation.

One possible representation corresponding with the requirement presented before is:

{ El_TDMS0
{{ sistema_ncms000 AEROS_np00000}_grup-nom-ms }_sn
{ controlará_vmif3s0 }_grup-verb
el_TDMS0
{{ ordenador_ncms000 }_grup-nom-ms}_sn
{ de_sps00 }_prep
a_bordo_RG000 y_CC00 el_TDMS0
{{ estado_ncms000
{ del_spcms { vehículo_ncms000 }_grup-nom-ms }_sp-de }_grup-nom-ms }_sn
espacial_AQ0CS00 ._Fp }_S

It is possible to obtain more than one syntactic-semantic representation from a requirement. This situation appears when the requirement contains some kind of ambiguity. For example, from our example, we can get two possible interpretations:

(1) Spanish: El sistema AEROS controlará el ordenador de a bordo del vehículo espacial y el sistema AEROS controlará el estado del vehículo espacial.
English: The AEROS system shall control the computer on board of the space-vehicle and the AEROS system shall control the status of the space-vehicle.
(2) Spanish: El sistema AEROS controlará el ordenador de a bordo y el sistema AEROS controlará el estado del vehículo espacial.
English: The AEROS system shall control the computer on board and the AEROS system shall control the status of the space-vehicle.

The ambiguity controller must identify the representation, which corresponds to the engineer's idea. It applies a series of rules to the set of semantic representations in order to qualify them. An example of a possible rule is the one, which asserts that the preposition *of* is always related only with the last nominal group (second interpretation). This controller can cooperate with the engineer to select the correct semantic representation.

From the syntactic-semantic representation the simplicity controller detects whether the structure of the sentence is simple or compound. If the requirement is composed, for instance, of two simple requirements, the validation process can continue dealing with these two in a sequential way.

Conceptual Refinement Module

The Conceptual Refinement Module receives the syntactic-semantic representation of a simple requirement and validates it in relation to the Requirements Base. The validation process is broken down into two steps: conceptual analysis and duplication control.

The conceptual analysis identifies, in the Knowledge Base, those entities involved in the syntactic-semantic representation, taking into account which kind of requirement has been introduced and its corresponding roles. The information contained in the Software Requirements Modeling component helps to identify the semantics roles required in each case. If all the objects and all the activities are present in

the Knowledge Base, the conceptual analysis constructs the requirement conceptual representation. If not, the system shows to the software engineer the entities (Objects or Activities) that should be added to the KB. From the following syntactic-semantic representation

```
{ El_TDMS0
{ { servicio_ncms000 }_grup-nom-ms }_sn
directorio_AQ0MS00
{ deberá_vmif3s0 { facilitar_vmn0000
}_infinitiu }_grup-verb
la_TDFS0
{ { conexión_ncfs000 }_grup-nom-fs }_sn
directa_AQ0FS00 ._Fp }_S
```

corresponding to the functional requirement:

Spanish: El servicio directorio deberá facilitar la conexión directa.

English: The directory service should facilitate the direct connection.

the conceptual analysis identifies servicio_directorio as the agent, deber_facilitar as the action and conexión_directa as the Patient.

After that, the duplication controller will verify that the requirement introduced by the software engineer contains new information. To do so, it matches the present requirement conceptual representation with the Requirements Base, taking into account which kind of requirement has been introduced, in order to discover possible duplications. If the present requirement conveys new information its conceptual representation could be added to the Requirements Base depending on the Software Quality Control Module.

In Figure 4 we can see two Requirement Conceptual Representations corresponding with two functional requirements:

Spanish: (req 8) El servicio directorio deberá facilitar la conexión directa.
Spanish: (req 9) El servicio directorio deberá facilitar el almacenamiento de referencia.

English: (req 8) The directory service should facilitate the direct connection.
English: (req 9) The directory service should facilitate the reference storage.

The original requirement was a compound requirement "The directory service should facilitate the direct connection and the reference storage." that has been decomposed in two simple requirements by the simplicity controller.

Software Quality Control Module

This module carries out a series of optional analyses, which validate the global Requirements Base incremented with the new requirement. The goal is to offer information about the software quality properties (completeness, traceability, consistency, verifiability and modifiability). Once a requirement has been validated and the information about software quality has been presented to the engineer, he decides if the requirement conceptual representation must be added to the Requirements Base.

Taking into account that these analyses are optional, the engineer could decide not to use them (it is a normal decision when adding the first requirements). In this case, these modules can be further used to control the global quality software specification. The analyses related to the quality properties already studied are described in the following paragraphs.
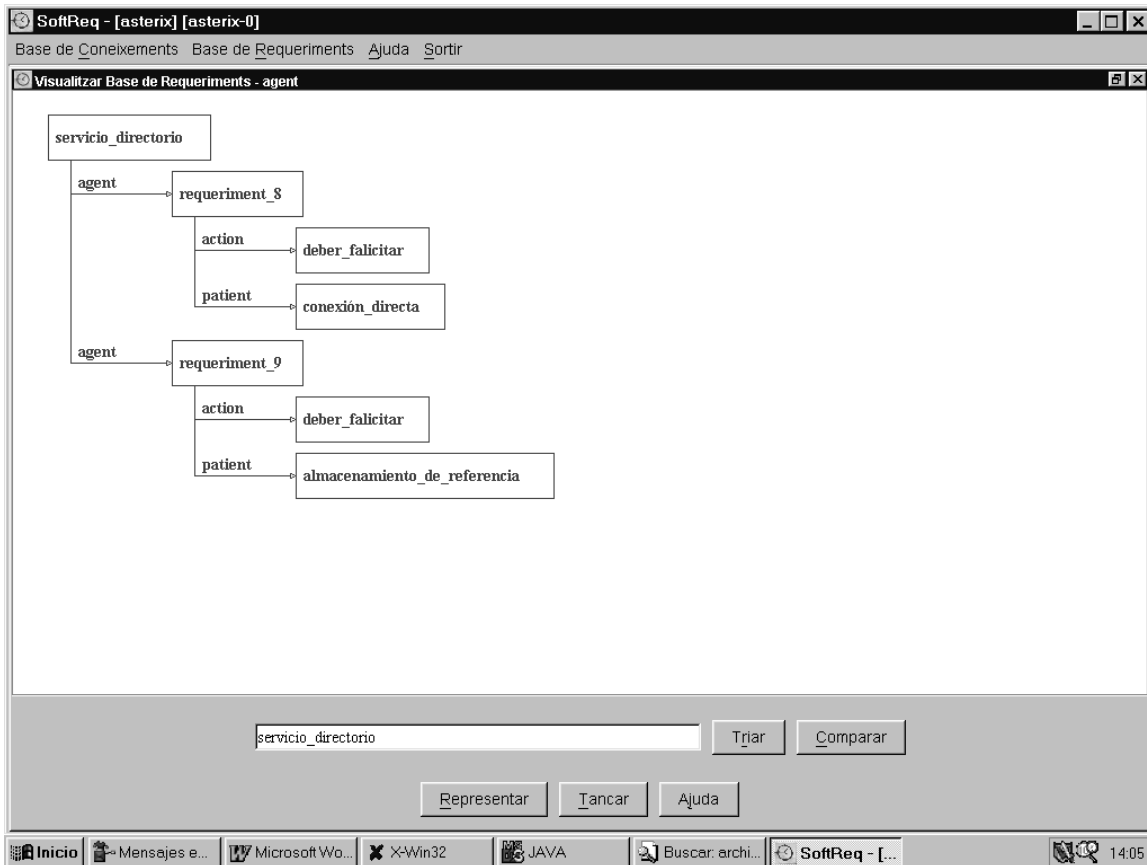
Figure 4. Requirements Conceptual Representation

Analysis of Completeness

To detect incomplete specifications, the system will activate reasoning mechanisms, taking into account several aspects of completeness. In order to control completeness, it is necessary to have available a general hierarchy of actions-subactions that can be associated to any set of specifications. An example of this hierarchy is the activity monitor, which comprises three subactivities: receive, analyze and display. Given the requirement:

"The ALPHA system will analyze and display the status of the space vehicle"

and taking into account the hierarchy described above, the system analyses the relationships among requirements in the Requirements Base. If there is no requirement containing the monitor activity, it will inform the engineer. In the same way, if the system notices that there is a requirement that contains the monitor activity but there is no requirement containing its subactivities, it will inform the engineer that the current specification could be incomplete.

Analysis of Traceability

The goal, for this quality property, is to provide the engineer with information about the traceability links of the requirement. The system will activate a set of existing algorithms, which control the traceability in order to show the relationships between the introduced requirement and a subset of requirements of the RB (either more specific or more general). From this information the engineer can see the relationships between requirements introduced up to this point. From the requirement:

"The ALPHA system will receive the data of the space vehicle"

the system will display, among others, the following more general requirement:

"The ALPHA system will monitor the data of the space vehicle"

The second requirement is more general than the first because *receive* is part-of *monitor*.

A lack of links shows that the requirement is isolated in relation to the rest of requirements.

Analysis of Modifiability

The complexity and consistency of future modifications of software specifications depend on the level of propagation of a given modification in all requirements affected by that modification. The concept of modifiability was formalized using software metrics, in particular we use the level of interconnection between the requirements. The range of possible values is [0..1], but the range of acceptable values is a subinterval of this. In case the obtained measure does not fall within this second range, the engineer must study a possible problem of excessive or insufficient interconnection.

Vertical and Horizontal Processing

The original goal of SAREL system, to assist engineers in the creation of SRS document, has been extended to analyze the correspondence between two SRS documents. Therefore, two operational modes must be distinguished. Figure 5 shows the modules used in Vertical and Horizontal Processing.
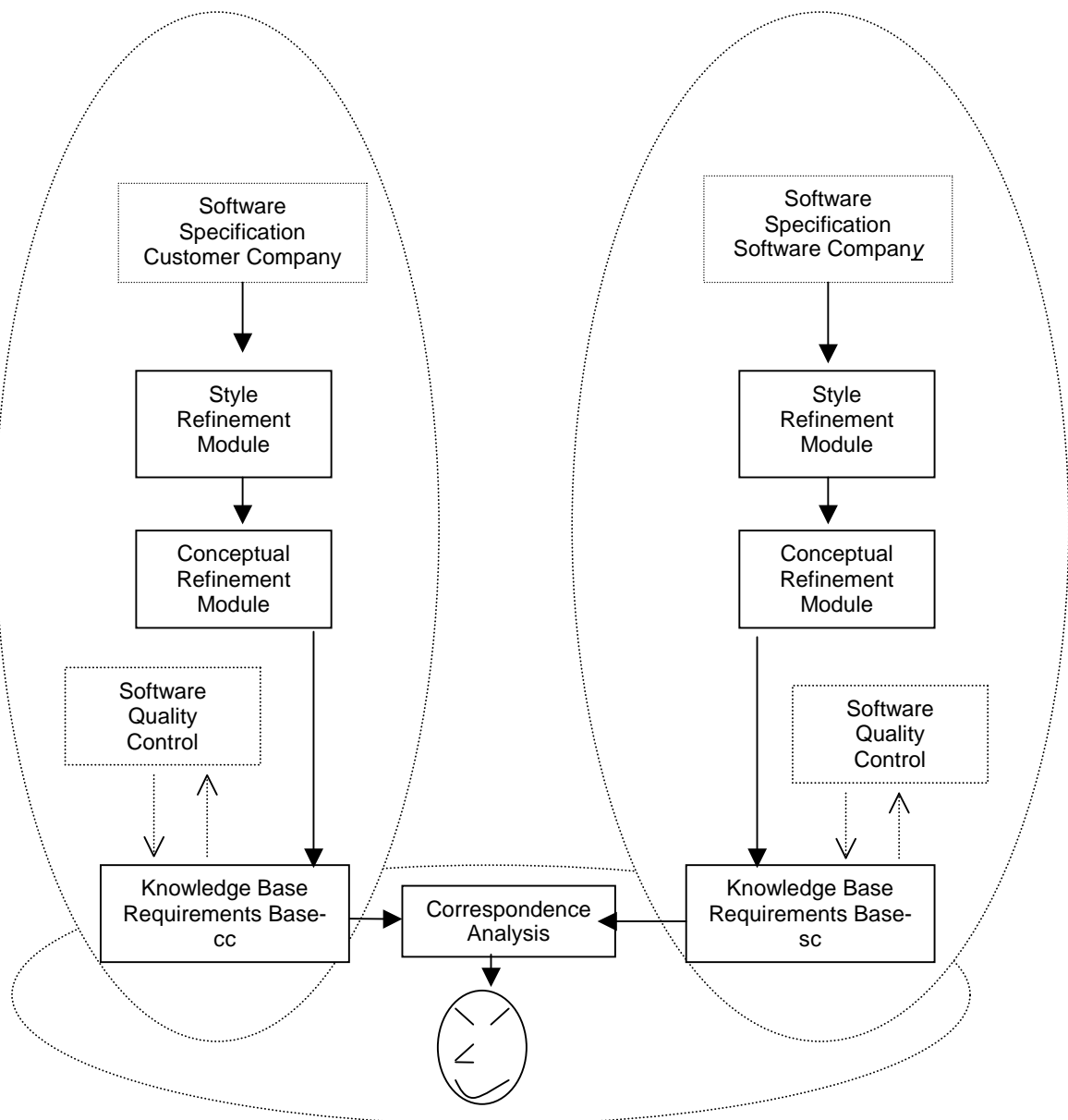

Figure 5. Vertical and Horizontal Processing

In the Vertical Processing, the input is a software specification written in natural language and the output is the Conceptual Representation associated that optionally can be manipulated by the engineer using the Software Quality Control Module. This process validates every requirement introduced by the engineer using the Style Refinement Module and the Conceptual Refinement Module. At the end the issue is the Conceptual Representation corresponding to all the information contained in the software specification. Using the SAREL system, the engineers can share, in a more reliable format, the information contained in the Knowledge Base and in the Requirements Base. Once the conceptual representation has been obtained, it can be globally validated using the optional analysis contained in the Software Quality Control Module.

In the Horizontal Processing, the input consists of two different conceptual representations, and the goal here is to offer information about the correspondence between them. As a first attempt, the correspondence analysis searches for every requirement1 in document1 its corresponding requirement2 in document2. Where document1 corresponds with the User Company that needs to develop a computer system and therefore, document2 corresponds with the Software Company that will carry this out.

During the correspondence search it is useful to take into account information about what kind of requirement requirement1 is (Functional, Performance or External Interface). In this case the system will give a correspondence measure, based on similarity analyses applied over the components of the requirements. Depending on the value of this measure, the correspondence will be tagged as Correct, Excess or Excess-Insufficient. This corresponding analysis will be improved giving consideration to all the concepts contained in both Requirements Bases instead of individually looking at the requirements.

As natural language is the best way of human communication, the existence of systems that are able to deal with are truly appreciate. When developing complex systems, software engineers can take a lot of profit of a tool like SAREL. They can write the specifications using natural language and the system will analyze and represent the contents. Instead of managing a huge amount of document, the access to the knowledge representation of the specifications offers a new way of interaction between engineers, the possibility of explore the conceptual meaning of the specifications, and the development of tools to control some aspects that are very tedious or even impossible to carry out in the original documents. On the other hand, the misunderstandings between customers and developers can be avoided with conceptual tools like SAREL. The matching between the representation of two documents can be more objective and useful than the matching between two texts in natural language. Therefore the advances in the linguistic engineering and knowledge engineering areas can be fruitfully applied to the software engineering area.

Acknowledgements

References

[1] Karl E. Wiegers R. "Software Requirements" Microsoft Pres 1999.

[2] R. Melchisedech, "Investigation of Requirements Documents Written in Natural Language", Requirements Engineering (1998) 3:2, pp. 91-97.

[3] C. Ben Achour, "Linguistic Instruments for the Integration of Scenarios in Requirements Engineering", Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'97)", Barcelona, Catalonia, Spain, June, 1997, pp. 93-106.

[4] E. Fuchs, and R. Schwitter, "Attempto Controlled English (ACE)", First International Workshop On Controlled Language Applications. Katholieke Universiteit Leuven, Belgium 1996.

[5] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, "An Automatic Quality Evaluation for Natural Language Requirements", Proceedings of the Seventh International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'01), June 4-5, 2001, Interlaken, Switzerland 2001.

[6] IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications, 1998.

[7]] J. Atserias, S. Climent, X. Farreres, G. Rigau, and H. Rodríguez. Combining Multiple Methods for the Automatic Construction of Multilingual WordNets. In Procceeding of RANLP'97 Bulgaria. 1997.

[8] J. Atserias, J. Carmona, I. Castellón, S. Cervell, M. Civit, L. Màrquez, M.A. Martí, L. Padró, R. Placer, H. Rodríguez, M. Taulé &; J. Turmo. Morphosyntactic Analysis and Parsing of Unrestricted Spanish Text. First International Conference on Language Resources and Evaluation (LREC'98). Granada, Spain, 1998.

[9] L. Padró, POS Tagging Using Relaxation Labelling Proceeding of COLING 96, Copenhagen Denmark, 1996.

[10] Castell, N. & Hernández, A., The Software Requirements Modeling in SAREL. Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'98), pp 49-56, Pisa, Italy, 1998.